Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Interaktivní robotický systém s implementací základů umělé inteligence

Interactive Robotic System with Artificial Inteligence Implementation

2018 **Radek Petráš**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **Radek Petráš**

Studijní program: B2649 Elektrotechnika

Studijní obor: 2612R041 Řídicí a informační systémy

Téma: Interaktivní robotický systém s implementací základů umělé inteligence
Interactive Robotic System with Artificial Inteligence Implementation

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Rozbor problematiky základních principů a způsobů interakce a rozbor robotických systémů a základů inteligence.
2. Návrh interaktivního robotického systému.
3. Realizace interaktivního robotického systému.
4. Verifikace a optimalizace interaktivního robotického systému.
5. Srovnání naměřených výsledků s teoretickými předpoklady.
6. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] SZELISKI, Richard. *Computer Vision: Algorithms and Applications.* London: Springer, 2010. Texts in computer science. ISBN 978-1-84882-935-0.
[2] PROKOP, Jiří. *Algoritmy v jazyku C a C++.* 3., aktualizované a rozšířené vydání. Praha: Grada, 2015. ISBN 978-80-247-5467-3.
[3] DATTA, Asit Kumar, Madhura DATTA a Pradipta Kumar BANERJEE. *Face detection and recognition: theory and practice.* Boca Raton: CRC Press/Taylor, 2016. ISBN 978-1482226546.
[4] HÁJOVSKÝ, Radovan, Radka PUSTKOVÁ a František KUTÁLEK. *Zpracování obrazu v měřící a řídící technice: učební text.* Ostrava: VŠB - Technická univerzita, 2012. ISBN 978-80-248-2596-0.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Zdeněk Macháček, Ph.D.**

Datum zadání: 01.09.2017
Datum odevzdání: 30.04.2018

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

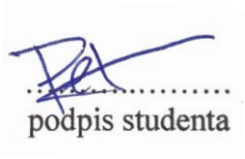
doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *28. dubna 2018*



Poděkování

Rád bych poděkoval Ing. Zdeňku Macháčkovi, Ph.D. za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

Abstrakt

Práce se zabývá návrhem interaktivního robotického systému s implementací několika druhů umělé inteligence. Práce popisuje již známé technologie a algoritmy potřebné pro návrh a pokračuje jejich úpravou a implementací. Na závěr jsou popsány možnosti další optimalizace a zrychlení.

Klíčová slova

Umělá inteligence; robotický systém; interaktivita; rozpoznání obličeje; hra;

Abstract

The thesis deals with the design of an interactive robotic system with implementation of several types of artificial intelligence. The thesis describes the already known technologies and algorithms needed for design and continues with their modification and implementation. Finally, the options for further optimization and acceleration are described.

Key words

Artificial Intelligence; robotic system; interactivity; face recognition; game;

Obsah

Seznam použitých zkratek a symbolů ..... - 7 -
Seznam obrázků ..... - 8 -
Seznam tabulek ..... - 9 -
Úvod ..... - 10 -
1 Rozbor problematiky ..... - 11 -
1.1 Robotický systém ..... - 11 -
1.2 Interaktivita ..... - 13 -
1.3 Umělá inteligence ..... - 13 -
2 Návrh robotického systému ..... - 16 -
2.1 Použitý hardware ..... - 16 -
2.2 Návrh softwaru ..... - 17 -
2.3 Detekce obličeje ..... - 17 -
2.3.1 Předzpracování obrazu ..... - 17 -
2.3.2 Obraz v odstínech šedi ..... - 17 -
2.3.3 Filtrování obrazu ..... - 18 -
2.3.4 Detektor Viola-Jones ..... - 19 -
2.3.5 Integrální obraz ..... - 19 -
2.3.6 Haarovy příznaky ..... - 20 -
2.4 Použití umělé inteligence ..... - 21 -
2.4.1 Fuzzy logika ..... - 21 -
2.4.2 Expertní systém ..... - 22 -
2.4.3 Strojové učení ..... - 23 -
3 Realizace robotického systému ..... - 24 -
3.1 Struktura ..... - 24 -
3.2 Realizace knihovny pro obsluhu kamery ..... - 24 -
3.3 Realizace vizualizační aplikace ..... - 26 -
3.3.1 Realizace hlavního menu ..... - 28 -
3.3.2 Realizace hry „Kvíz“ ..... - 28 -
3.3.3 Realizace hry „Fotbal“ ..... - 30 -

3.3.4 Realizace hry „Uhádnu dopravní prostředek“ ..... - 31 -
4 Verifikace a optimalizace ..... - 33 -
4.1 Verifikace detekce ..... - 33 -
4.2 Optimalizace programu ..... - 33 -
5 Srovnání výsledků ..... - 34 -
Závěr ..... - 35 -
Použitá literatura ..... - 36 -
Seznam příloh I
Seznam příloh na DVD II

Seznam použitých zkratek a symbolů

Zkratka	Význam
AI	Umělá inteligence
WPF	Windows Presentation Foundation
GUI	Grafické uživatelské rozhraní
V-J	Viola-Jones
FPS	Snímků za sekundu (anglicky Frames per second)
RAM	Random Access Memory
RGB	Červená-modrá-zelená (barevný prostor)
XAML	Extensible Application Markup Language
ID	Identifikační číslo
HW	Hardware

Seznam obrázků

Obrázek 1.1: Zjednodušené blokové schéma obecného robotického systému ..... - 12 -
Obrázek 1.2: Zjednodušený model umělého neuronu ..... - 14 -
Obrázek 1.3: Model umělé neuronové sítě [8] ..... - 15 -
Obrázek 2.1: Hardware robotického systému ..... - 16 -
Obrázek 2.2: Barevný obraz a jeho ekvivalent v odstínech šedi ..... - 18 -
Obrázek 2.3: Základní konvoluční masky ..... - 18 -
Obrázek 2.4: Vlevo zdrojový obraz v odstínech šedi, uprostřed obraz po aplikaci lineárního čtvercového filtru 3x3 a vpravo obraz po aplikaci bilineárního filtru ..... - 19 -
Obrázek 2.5: Výpočet plochy pomocí integrálního obrazu ..... - 19 -
Obrázek 2.6: Originální obraz (vlevo) a jeho integrální obraz (vpravo) ..... - 20 -
Obrázek 2.7: Hranové příznaky ..... - 20 -
Obrázek 2.8: Čárové příznaky ..... - 21 -
Obrázek 2.9: Diagonální příznaky ..... - 21 -
Obrázek 2.10: Funkce příslušnosti pro změnu rychlosti ..... - 22 -
Obrázek 2.11: Funkce příslušnosti pro změnu jedné ze souřadnic ..... - 22 -
Obrázek 2.12: Princip fungování expertního systému ..... - 23 -
Obrázek 3.1: Třídní diagram knihovny ..... - 25 -
Obrázek 3.2: Blokové schéma metody Detect ..... - 25 -
Obrázek 3.3: Stavový diagram aplikace ..... - 27 -
Obrázek 3.4: Zjednodušený vývojový diagram aplikace ..... - 27 -
Obrázek 3.5: Hlavní menu ..... - 28 -
Obrázek 3.6: Hra „Kvíz“ ..... - 29 -
Obrázek 3.7: Hra „Fotbal“ ..... - 31 -
Obrázek 3.8: Hra „Uhádnu dopravní prostředek“ ..... - 31 -
Obrázek 3.9: Znalostní báze hry ..... - 32 -
Obrázek 3.10: Verifikace detekce ..... - 33 -

Seznam tabulek

Tabulka 5. Srovnání detekčních dob ..... - 34 -

Úvod

S rozvojem výpočetní techniky se dnes čím dál více setkáváme s robotickými systémy. Tyto systémy můžeme potkat v nemocnicích, průmyslu, školství, domácnostech a na mnoha dalších místech. Kdysi jednoduché mechanické stroje se mění na velice sofistikovaná mechatronická zařízení řízená počítačem, která jsou schopna v mnoho případech zcela nahradit člověka. Díky tomu jsou kladeny vyšší požadavky na bezpečnost těchto systémů a jejich multifunkčnost. Dnešním trendem je implementování určitého druhu umělé inteligence a strojového učení do robotického systému, který je schopen se díky tomu učit a reagovat na nové druhy situací.

Tato práce se zaměřuje na robotický systém reagující na člověka. Pomocí kamery a open-source knihovny OpenCV bude detekován obličej člověka stojícího před robotickým systémem. Implementace této části bude provedena v C++. Dále dojde ke spuštění interaktivní hry a algoritmu AI, který bude reagovat na uživatele ovládajícího robotický systém. Implementace této části bude provedena v jazyku C# s použitím WPF frameworku pro vykreslení GUI.

Na začátku práce jsou rozebrány základní informace o robotických systémech, způsobech interakce a umělé inteligence. Jsou zde popsány základní komponenty robotických systémů a stručně shrnuto, co je třeba zohlednit při návrhu takového systému. Následně jsou rozebrány možnosti interakce mezi robotickým systémem a jejich příklady. Konec kapitoly se věnuje umělé inteligenci, definuje, co vše lze považovat za umělou inteligenci a zabývá se nejpoužívanějšími typy umělé inteligence.

Další část práce se zabývá návrhem systému. Na začátku kapitoly je obecně rozebrán hardware a software robotického systému. Součástí návrhu je popis matematických základů vybraných algoritmů a jejich částí. Dále jsou zde rozebrány jednotlivé návrhy implementací umělé inteligence a jejich význam pro systém.

Následující kapitola popisuje realizaci softwaru robotického systému. Podkapitoly popisují všechny funkce knihovny a jednotlivá vizualizační okna. Součásti popisu jsou zjednodušené vývojové diagramy a snímky z běhu programu.

Poslední kapitoly verifikují funkčnost hlavních částí programu a detekce obličeje. Součástí je taktéž informace o optimalizaci programu jak během vývoje, tak i po jeho ukončení. Následně jsou porovnány výsledky s referenčním systémem, na kterém běží shodná aplikace.

V závěru jsou stručně shrnuty dosažené výsledky, možnosti dalšího rozvoje, jak hardwaru robotických systémů, tak i jejich softwarové výbavy.

1 Rozbor problematiky

1.1 Robotický systém

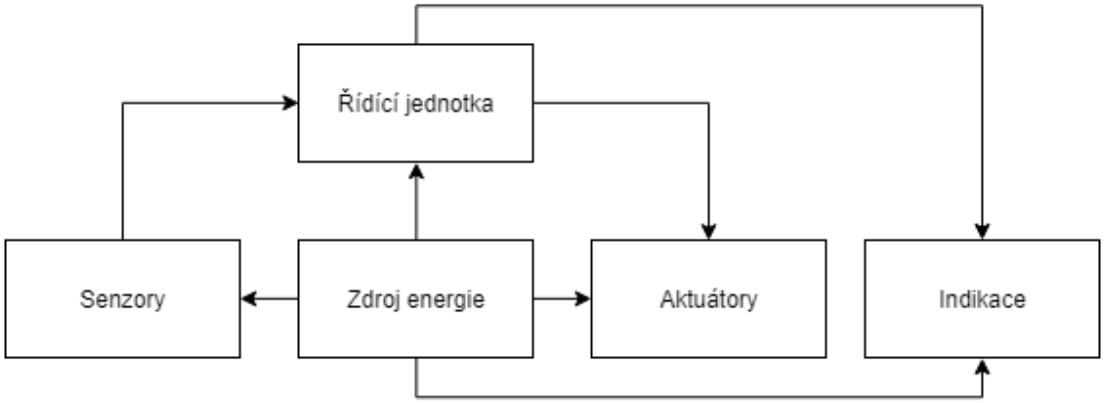
Robotickým systémem rozumíme elektrotechnicky mechanický stroj řízený digitálním softwarem určený k interakci s člověkem, k zjednodušení lidských činností anebo úplnému nahrazení člověka v určitých procesech výroby, manuálně náročných činnostech a podobných činnostech. Robotické systémy se skládají z množství různých komponentů.

Robotické systémy můžeme rozdělit do několika kategorií. První a asi nejrozšířenější kategorií jsou robotické systémy určené pro průmysl. Jedná se o podavače, robotická ramena a podobné robotické automaty. Tyto robotické systémy se používají většinou na přesun materiálu v továrních komplexech. Výhodou je, že průmyslové robotické systémy dokáží zvednout stovky kilogramů, rychle a přesně s nimi manipulovat. Tyto systémy mohou být znovu naprogramovány, ve většině případů se ale robotický systém jednou naprogramuje a tento program v něm zůstává až do jeho odstavení. [1]

Druhou kategorií jsou robotické systémy určené pro výzkum. Tyto systémy jsou většinou mnohem složitější než systémy pro průmysl. Zatímco robotické systémy v průmyslu mívají pouze 6 stupňů volnosti, robotické systémy pro výzkum mívají stupňů více. Stupněm volnosti rozumíme možnost pohybu v určité ose. Větší počet stupňů volnosti umožňuje robotickému systému lepší práci s předměty, lépe simuluje lidské ruce. Například dokáže velice jednoduše dosáhnou za nějakou překážku. Tyto robotické systémy dokáží v mnoha ohledech simulovat člověka a jeho úkony jako chůzi, manipulaci vyžadující jemnou motoriku apod. [2]

Poslední kategorií robotických systému jsou systémy určené pro zábavu. Do této kategorie spadají všechny robotické systémy nezařazené do předchozí kategorie. Tyto robotické systémy mohou být ve formě jednoduchého ramena na zábavních atrakcích až po dnešní hračky, které simulují zvířecí společníky. Tyto systémy slouží pro pobavení a usnadnění práce svým majitelům. [3]

Robotické systémy lze ale rozdělit i podle jiných kritérií. Například podle podmínek, kde je robotický systém schopen fungovat, se robotické systémy dají rozdělit na systémy pro práci pod vodou, systémy pro práci v prašném prostředí, systémy pro práci v extrémních teplotách, systémy pro práci ve vesmíru.



Obrázek 1.1: Zjednodušené blokové schéma obecného robotického systému

Robotický systém se skládá z množství komponent (obrázek 1.1). Nejdůležitějšími komponenty jsou pro robotický systém jsou zdroje energie a řídící jednotka. Bez těchto dvou komponent robotický systém nemůže fungovat. Následně jsou zde přítomny prvky pro vnímání okolí a pro pohyb robota. Části pro přeměnu elektrického signálu na mechanickou energii nazýváme aktuátory a součásti robota vnímající okolní prostředí a podmínky se nazývají senzory či snímače. Senzor na rozdíl od snímače obsahuje další elektronické obvody, které upravují, zesilují či převádí výstupní signál, zatímco snímač poskytuje je pouze součástka, která měřenou veličinu převede na elektrickou veličinu.

Základní součástí robotického systému je zdroj energie. V místech, kde nemůžeme zaručit neustálý přístup k rozvodné elektrické síti, musíme zajistit napájení systému z jiného zdroje. První možností jsou olověné akumulátory. Mezi jejich výhody patří bezpečnost a relativně dlouhá životnost. Oproti stříbrno-kadmiovým akumulátorům jsou ale mnohem těžší. Další možností je využití generátoru elektrické energie. Tato volba přináší řadu problémů s rozvody paliva a rozvody tepla. Při návrhu robotického systému napájeného baterií je vždy třeba uvážit několik faktorů a to bezpečnost, životnost a hmotnost celého systému. [4]

Další součástí robotických systémů jsou akční členy neboli aktuátory. Jedná se o mechatronická zařízení schopná převádět digitální a analogový signál na mechanickou energii. Jednou z nejrozšířenějších variant jsou elektromotory, které svým otáčením pohybují s částmi systému, druhými jsou pneumatické a hydraulické aktuátory – ať už písty nebo ventily a podobná zařízení.

Pokud má robotický systém vnímat okolí (což je potřeba vždy) je důležité, aby součástí jeho konstrukce byly příslušné senzory. Existuje mnoho různých druhů senzorů, díky kterým jsou robotické systémy schopny vnímat okolní svět, samy detekovat chyby nebo nedostatky systému a defekty částí systému.

Důležitou částí robotických systémů je řídící jednotka. Tu může tvořit programovatelný automat (většinou u průmyslových aplikací), klasický desktopový procesor a potřebné periferie,

u jednodušších systémů lze použít mikrokontrolér nebo čistě analogové řízení, což se ale v praxi nepoužívá.

Při návrhu robotických systému je třeba počítat se všemi riziky a problémy, které se mohou při provozu robota objevit. Jedním z největších problémů bývá přehřátí systému, kdy dojde k navýšení teploty nad mez, kterou systém dokáže vydržet bez újmy. V následku přehřátí dojde ke zničení součástek a nefunkčnosti celého systému. Tohoto problému se dá zbavit zvýšením chlazení systému, což ale vyústí v narůst spotřeby a zvýšení tepelného výkonu zařízení.

1.2 Interaktivita

Obecně interaktivitou rozumíme interakci uživatele a stroje. Uživatel dává stroji podněty (dotek, stisknutí tlačítka, hlasový příkaz) a stroj na tento podnět adekvátně zareaguje. Pro to, abychom mohli takový systém realizovat, je nutné zajistit vstup uživatelských podnětů.

Uživatelské podněty lze rozdělit podle druhu interakce. Prvním typem uživatelské interakce může být dotek nebo jiný fyzický kontakt s robotickým systémem. Uživatel pomocí stisknutí tlačítek, ovládání vizualizace nebo měny tlaku na určitou oblast robota dává systému příkazy. Tato forma interaktivity je jednou z nejjednodušších forem pro realizaci a pochopení člověkem. [5]

Dalším typem interakce je interakce zvuková. Uživatel pomocí mluvené řeči nebo formou příkazů určuje robotickému systému, co má dělat a robotický systém na to patřičně reaguje. Robotický systém po celou dobu provozu „naslouchá“ okolí a následně všechny zvuky analyzuje. Toto řešení je dosti náročné na algoritmizaci a výkon, ale s příchodem a rozvojem technologie neuronových sítí se tyto systémy začínají rozvíjet a rozšiřovat stále rychleji.

Za poslední typ interakce lze považovat interakci vizuální. Robotický systém vnímá své okolí pomocí kamery, soustavy kamer nebo čidel mapujících místnost. Díky již známým algoritmům pro rozpoznání částí těla a algoritmům strojového učení je robotický systém schopen rozpoznat přesnou pózu člověka a reagovat na ni. Díky dlouhodobé optimalizaci algoritmů k tomu určených a nárůstu výpočetního výkonu řídících jednotek je robotický systém schopen detekovat a reagovat na uživatelské gesto s minimální zpožděním téměř v reálném čase.

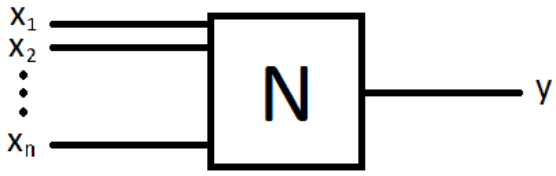
1.3 Umělá inteligence

Umělá inteligence je jedna ze součástí moderní informatiky, která se snaží u strojů a počítačových systémů vynutit inteligentní chování. Dodnes není přesně definován pojem inteligentního chování, většinou se inteligentní chování srovnává s lidským rozumem. Systémy umělé inteligence se vyznačují schopností uvažovat, plánovat následující kroky a učit se. Výhodou umělé inteligence je odstranění kognitivního zkreslení – zkreslení způsobené systematickou chybou, špatným rozhodnutím způsobeným na základě zkušenosti člověka. Bohužel umělá inteligence si může vyvinout vlastní kognitivní zkreslení.

Proto, abychom posoudili, zda je systém inteligentní či nikoliv se používá takzvaný Turignův test. Princip tohoto testu naznačil roku 1950 Alan Turing ve svém díle „Computing machinery and intelligence“[6]. Tento test říká, že systém je inteligentní, pokud uživatel nedokáže rozeznat výstup systému od výstupu živého člověka. Toto tvrzení se dá zpochybnit argumentem čínského pokoje. Tento argument říká, že pokud budeme mít systém, který není inteligentní, tzn. nepochopí uživatelem vloženou otázku, ale má dostatečně velkou databázi odpovědí na všechny možné otázky daného jazyka a odpověď v ní najde. Tento systém se uživateli jako inteligentní jevit, avšak ve své podstatě inteligentní není, protože nad problémem nijak „nepřemýšlí“.

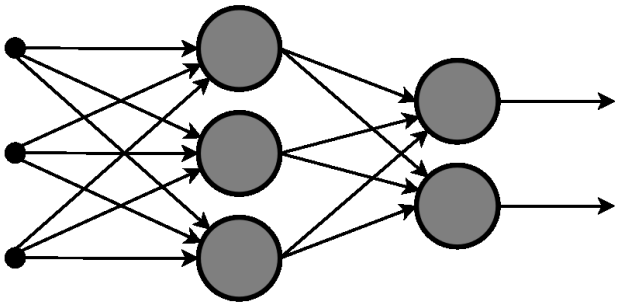
Inteligentní systémy se dají dělit na slabě inteligentní a silně inteligentní systémy. Slabě inteligentní systémy jsou ty, které sice projdou Turingovým testem, ale ve skutečnosti nejsou reálně inteligentní, silné jsou ty, které pouze nekopírují cizí řešení, ale vymyslí své řešení problému, tudíž nad problémem „přemýšlejí“.

Existuje několik způsobů, jak takový systém naprogramovat. Dnes asi nejpoužívanějším způsobem je využití neuronových sítí. Dnešní umělé neuronové sítě se inspirují v reálných biologických neuronových sítích. Tyto sítě jsou složeny z umělých neutronů. Umělý neutron má libovolný počet vstupů, ale pouze jeden výstup (obrázek 1.2). Na obrázku jsou vstupy označeny jako $x_1 – x_n$ a výstup označen jako y. Každý neuron se chová jako samostatný výpočetní uzel, který předává své výsledky a data dalším neuronům. V biologické síti se zkušenosti uchovávají pomocí dendritech. Umělá neuronová síť toto simuluje pomocí vah. [7]



Obrázek 1.2: Zjednodušený model umělého neuronu

Problém neuronových sítí je, že pro okolí se tváří jako černá skříňka. Po zadání vstupních dat nemůžeme určit, jak systém reaguje uvnitř. Známe tedy pouze výsledek výpočtu systému. Pokud se jedná o dostatečně jednoduchý úkol, který jsme si schopni ověřit vlastním výpočtem, můžeme zkontrolovat správnost umělé sítě. Pokud ale zadáme síti složitější problém, který nejsme schopni sami vyřešit, musíme pouze slepě věřit výsledku, který síť vypočte. Vzhledem k rapidnímu vývoji se předpokládá, že do 100 let umělá inteligence překoná naše myšlení ve všech směrech. Z těchto dvou důvodů se vývoj neuronových sítí snaží cílit směrem k vysvětlitelné AI.



Obrázek 1.3: Model umělé neuronové sítě [8]

Dalším způsobem programování patří genetické programování, které se snaží napodobit evoluci živočišných druhů. Program se snaží upravit svůj vlastní kód, tak aby byl co nejefektivnější pro určitou úlohu oproti své vlastní minulé verzi.

Pro naprogramování AI, která řeší úlohy, u kterých je možno definovat stav se používá metoda prohledávání stavových prostorů. Pro algoritmus definujeme počáteční a cílový stav. Celý problém si můžeme představit jako orientovaný graf, kde jednotlivé stavy představují uzly a přechody mezi nimi akce potřebné ke změně stavu. I přes vysoký výkon a paměťovou kapacitu dnešních počítačů je stále tento typ výpočtů stále pomalý, protože stavové prostory mohou být velice rozsáhlé, v některých případech i nekonečné. Počítače v těchto případech nemusejí dojít k přesnému řešení úlohy, ale pouze k částečné cestě mezi stavy grafu. Dalším problémem je někdy nejasný cílový stav. Jedná se o případy, kdy je známa pouze představa toho, jak by cílový stav mohl vypadat. [9]

Posledním způsobem jsou algoritmy strojového učení. Tyto algoritmy umožňují systému se „učit“ a tím zlepšovat svou efektivitu v řešení daného problému. Učením systému rozumíme změnu vnitřních stavů, které napomáhají lepšímu řešení problému. Strojové učení využívá metod statistiky a analýzy dat. Dnes nachází využití v rozpoznání řeči, textu, detekci podivného chování (nelegální užití platebních karet, hledání podezřelých osob v davu apod.). Využívají se dva hlavní typy strojového učení – učení s učitelem a učení bez učitele.

Strojové učení s učitelem používá soubor trénovacích dat, který je tvořen uspořádanými dvojicemi vstupních a výstupních dat. Systém se naučí na této testovací sadě a poté pracuje s neznámými daty. V tomto případě může dojít k přeučení systému – tzn. k přílišnému zaměření určitým směrem a neschopnosti správně reagovat na některé změny vstupních dat. Abychom se zbavili tohoto problému, můžeme použít většího objemu trénovacích data, aby výsledný systém byl schopen správně reagovat i na nepředpokládaná vstupní data. [10]

2 Návrh robotického systému

Cílem návrhu je vytvoření systému, který umožňuje uživateli nestandartní způsob ovládání (interakce) pomocí pohybu těla. Systém je navržen tak, aby detekoval uživatelův obličej a ten použil pro ovládání jednotlivých her. Druhou možnou interakcí je využití dotykového displeje, který je součástí robotického systému.

2.1 Použitý hardware

Software je aplikován na již existující hardware a to robotický systém Advee. Základem tohoto systému je hliníková konstrukce s laminátovým krytem. Uvnitř robota je umístěn audio zesilovač Renkforce, k němuž jsou připojeny 2x20W reproduktory pro ozvučení celého systému. Pro interakci s uživatelem je zde dotykový monitor o rozlišení 1024x1280 pixelů a kamera Microsoft® LifeCam Cinema o maximálním rozlišení 1280x720@30FPS. Vyjma monitoru se pro výstup k uživateli dá využít tepelná tiskárna Zebra model TTP 7030 pro tisk na papírovou roli šířky 80mm, ze které uřízne požadovaný kus. Výpočetní výkon je poskytnut standartním stolním počítačem. Počítač je postaven na základové desce ASRock E350M1 formátu Mini-ITX. Deska je osazena 64 bitovým procesorem AMD E-350. Tento procesor se skládá ze dvou výpočetních jader běžících na frekvenci 1,6 GHz. Počítač má k dispozici dvě paměti RAM o velikosti 2GB, celkově tedy 4 GB operační paměti. Na obrázku 2.1 vidíme vnitřní uspořádání hardwaru (vlevo) a vnější, které slouží k interakci s uživatelem (vpravo).

Obrázek 2.1: Hardware robotického systému

2.2 Návrh softwaru

Na použitém počítači je nainstalován 32 bitový operační systém Windows 7 Home Premium Service Pack 1. Software robotického systému je navrhován jako interaktivní hra. Uživatel si nejprve zvolí hru a následně pomocí pohybu hlavy danou hru ovládá. V softwaru je realizováno několik her, aby si uživatel hru mohl zvolit. Součástí softwaru je implementace několika druhů umělé inteligence.

Software je navržen jako několik paralelních vláken. V prvním vlákně se získávají data z kamery, následně se provede předzpracování obrazu pro detektor a dojde k použití detektoru obličeje. Výsledná data se předají do hlavního vlákna. Hlavní vlákno slouží k obsluze vizualizace a vykreslování rozhraní, zachytávání událostí způsobených uživatelem a GUI. Vždy při spuštění hry se oddělí třetí vlákno starající se o herní pravidla a hru jako takovou. Po dokončení hry dojde k ukončení třetího vlákna zobrazení vyhodnocení a přechodu zpět do menu.

2.3 Detekce obličeje

Detekce obličeje je použita jako hlavní prvek interakce robotického systému a člověka. Pomocí pozice detekovaného obličeje v zorném poli kamery dochází k ovládání dané hry, tzn. posunu kurzoru po herní ploše.

2.3.1 Předzpracování obrazu

Pro mnou používané algoritmy pracujeme v barevném prostoru RGB. Většina dnešních algoritmů pracuje s obrazem tvořeným stupni šedi – výhodou v tomto případě je snížené množství výpočtů. Z třech výpočtů pro každou barevnou složku se počet sníží na pouze jeden výpočet a tím dojde k úspoře času a výkonu a tím možnosti běhu algoritmů vyšší rychlostí, skoro v reálném čase.

2.3.2 Obraz v odstínech šedi

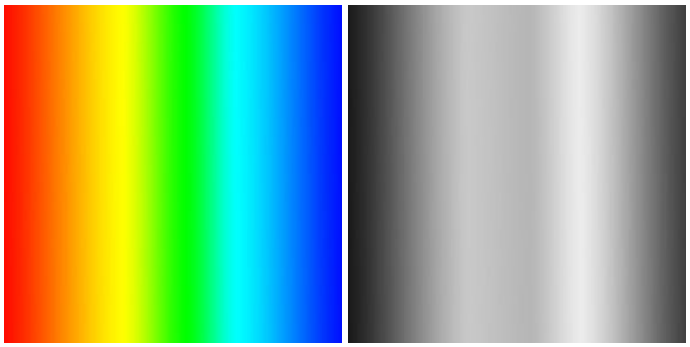
Prvním krokem předzpracování je převod z RGB barevného obrázku na obrázek ve stupních šedi. Toto je docíleno pomocí jednoduchého matematického vzorce (rovnice 2.1), kdy je každé složce R, G a B určena váha, kterou působí na vytvoření úrovně šedi S

$$S_{(x,y)} = R_{(x,y)} \times V_R + G_{(x,y)} \times V_G + B_{(x,y)} \times V_B \tag{2.1}$$

kde V_R je váha červené složky, V_G váha složky zelené a V_B váha modré složky. Součet těchto vah musí být roven maximálně jedné. Pro zachování stejné svítivosti obrazu v úrovních šedi vůči obrazu barevnému byly zvoleny parametry [11]

$$V_R = 0{,}2125;\ V_G = 0{,}7154;\ V_B = 0{,}0721 \tag{2.2}$$

z čehož je patrné, že největší podíl na výsledném stupni šedi má zelená složka obrazu.

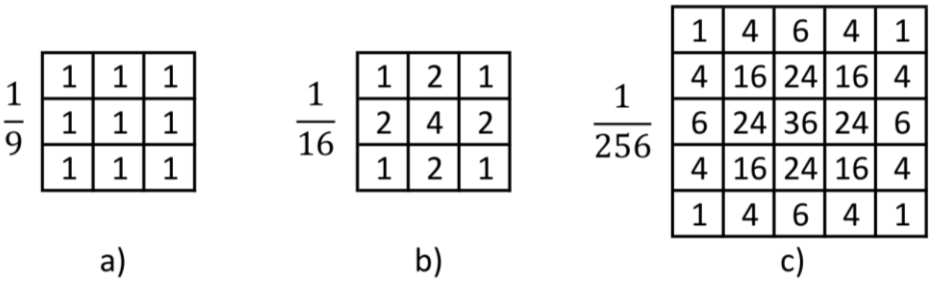


Obrázek 2.2: Barevný obraz a jeho ekvivalent v odstínech šedi

2.3.3 Filtrování obrazu

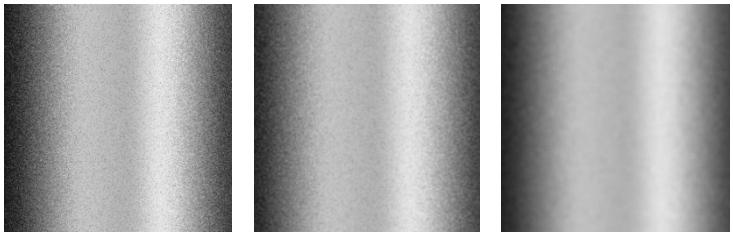
Dalším krokem při úpravě obrazu před samotnou detekcí je odstranění vysokofrekvenčního šumu v obraze. Nejjednodušeji toho lze docílit lineárním filtrováním tzn. aplikací konvoluční masky. Výsledná hodnota obrazového bodu je pak určena jako součin konvoluční masky a odpovídající části původního obrazu.

Mezi základní konvoluční masky patří tyto masky (obrázek 2.3):



Obrázek 2.3: Základní konvoluční masky

Na obrázku 2.3 a) vidíme jednoduchou čtvercovou masku, která průměruje okolí obrazového bodu. Všechny body mají stejnou váhu, tudíž při velkých změnách svítivosti (například pruh vysoké svítivosti obklopený dvěma tmavými) dojde ke ztrátě této informace, což může být v mnoha případech problém. To z části řeší maska b), která zobrazuje bilineární filtr. Tento filtr dává vysokou váhu středovému bodu, čímž se zachová dostatečně jeho informace a zároveň také zvyšuje váhu jak horizontálního, tak vertikálního středového řádku masky. Tím dojde k zachování vodorovných a horizontálních hran v obraze. Poslední maskou je Gaussova maska c), která využívá k výpočtu větší okolí obrazového bodu. Srovnání výstupu konvolučních masek poskytuje obrázek 2.4.



Obrázek 2.4: Vlevo zdrojový obraz v odstínech šedi, uprostřed obraz po aplikaci lineárního čtvercového filtru 3x3 a vpravo obraz po aplikaci bilineárního filtru

Při použití konvolučních masek nastává problém u hran obrazu. Protože výpočet používá okolí bodu, které ale v těchto místech není kompletní, používá se několik způsobů rozšíření obrazu. Zde nachází použití různé metody doplnění obrazu jako je kopírování, zrcadlení, roztažení apod.

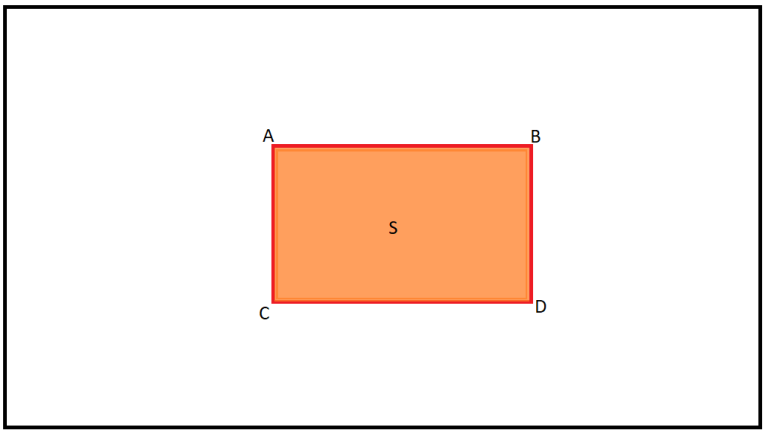
2.3.4 Detektor Viola-Jones

Pro samotnou detekci obličeje v obraze je použit Viola-Jones (dále V-J) detektor představený roku 2001 pány P. Violou a M. Jonesem. Tento detektor se skládá z několika částí. Jako první detektor V-J vypočte integrální obraz jasového obrazu, následuje výpočet Haarovych příznaků pro několik vzorů a následná klasifikace pomocí AdaBoost algoritmu, který pomocí slabých klasifikátorů vytvoří klasifikátory silné za účelem zvýšení rychlosti a přesnosti detekce.

Mezi hlavní výhody V-J detektoru patří jeho rychlost, vysoká spolehlivost a dostatečná nezávislost na osvětlení a velikosti objektu.

2.3.5 Integrální obraz

Integrální obraz se používá při digitální zpracování obrazu k urychlení výpočtu. Integrální obraz určuje součet hodnot všech pixelů v oblasti bodu se souřadnicemi [0,0] a aktuálně zpracovaného bodu. Výhodou je že díky tomu jsme schopni určit součet jakékoliv oblasti obrazu pouze pomocí již vypočteného integrálního obrazů a prací s rohy námi určené oblasti.



Obrázek 2.5: Výpočet plochy pomocí integrálního obrazu

Výpočet oblasti vyznačené na obrázku 2.5 pomocí bodů A, B, C, D provedeme pomocí vzorce

$$S = H_D - H_C - H_B + H_A \tag{2.3}$$

kde H_A udává hodnotu integrálního obrazu v bodě A, H_B hodnota v bodě B, H_C hodnota v bodě C a H_D hodnota v bodě D. Z tohoto je patrná již zmíněná výhoda – namísto sčítání všech hodnot v celé oblasti dojde k jednoduchému výpočtu se čtyřmi body.

Samotný výpočet integrálního obrazu probíhá dle vzorce

$$I_{(x,y)} = i_{(x,y)} + I_{(x,y-1)} + I_{(x-1,y)} - I_{(x-1,y-1)} \tag{2.4}$$

kde *x* a *y* určují souřadnice obrazového bodu, *I* určuje vypočtený integrální obraz a *i* obraz původní. Z rovnice 2.4 je patrné, že integrální obraz lze jednoduše vypočítat pouze jedním průchodem původního obrazu. Na obrázku 2.6 vidíme příklad obrazu a jeho integrálního obrazu.

2	12	13
5	18	11
4	19	26

2	14	27
7	37	61
11	60	110

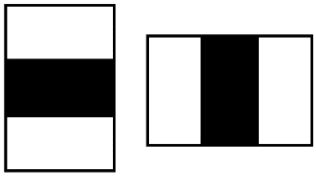
Obrázek 2.6: Originální obraz (vlevo) a jeho integrální obraz (vpravo)

2.3.6 **Haarovy příznaky**

Výpočet Haarových příznaků se dělí podle počtu zvolených oblastí. První typem příznaků je hranový příznak (obrázek 2.7). Jde o rozdíl sum dvou oblastí. Tento příznak se používá například k detekci očí – oblast očí je vždy tmavší oproti oblasti čela.

Obrázek 2.7: Hranové příznaky

Dalším typem příznaku jsou příznaky čárové. Tyto příznaky se skládají ze tří částí, přičemž výsledný příznak určíme jako rozdíl sumy vnějších částí a sumy části vnitřní. Tento příznak se používá pro nalezení oblasti nosu.



Obrázek 2.8: Čárové příznaky

Posledním používaným typem příznaku je příznak diagonální skládající se ze čtyř oblastí, kde je výsledný příznak vypočten jako rozdíl sum oblastí na diagonále.

Obrázek 2.9: Diagonální příznaky

Velikost jednotlivých příznaků se mění postupně od velikosti 1x1 pixel až po velikost obrázku. Pro vstupní obraz 3x3 pixely to je přibližně 40 příznaků. Z toho je patrné, že při zvyšování rozlišení bude počet příznaků rychle narůstat a s ním i výpočetní náročnost algoritmu. Z těchto příznaků jsou vybrány jen ty, jejichž hodnota by mohla odpovídat hodnotám určeným pro obličej.

Z tohoto souboru vhodných příznaků se pomocí klasifikačního algoritmu Adaboost vybere pouze část, která již odpovídá obličejům v obraze.

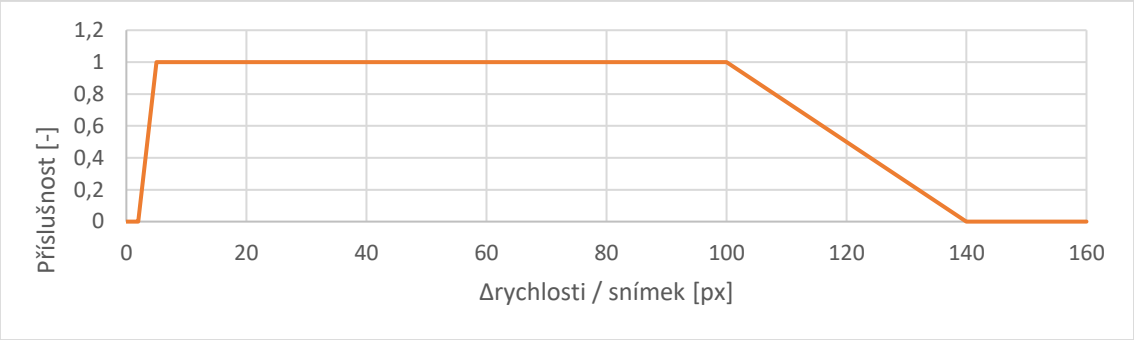
2.4 Použití umělé inteligence

Robotický systém obsahuje implementaci několik druhů umělé inteligence. Nejjednodušší formou umělé inteligence je použití fuzzy logiky, pro ošetření drobných nedostatků detekce obličeje. Další implementovanou formou umělé inteligence pro robotický systém je použití expertního systému jakožto rozhodčího pro hru „Uhádni dopravní prostředek“. Poslední formou použití umělé inteligence je využití strojového učení při detekci obličeje.

2.4.1 Fuzzy logika

Implementace fuzzy logiky se skládá z procesu fuzzyfikace, následně zpracování dat a zpětné defuzzyfikace. Na rozdíl od výrokové logiky, fuzzy logika nedefinuje pouze hodnoty 1 a 0 či pravda, nepravda, ale také definuje stavy mezi těmito hodnotami a přechody mezi nimi. Použití této logiky je vhodné tam, kde nejsou přesně definované hranice nebo se částečně prolínají např. horká, teplá a studená voda. V rámci implementace je fuzzy logiky použito pro ošetření chyb při detekci obličeje. Hlavním problémem při detekci obličeje je výskyt několika obličejů v obraze a zvolení toho správného. Abychom mohli detekovat správný obličej a udržet si informaci o jeho poloze po celou dobu hraní, je nutné zamezit chybné detekci ostatních obličejů. Pomocí výpočtu změny rychlosti pohybu obličeje mezi dvěma snímky jsem schopen určit, zda

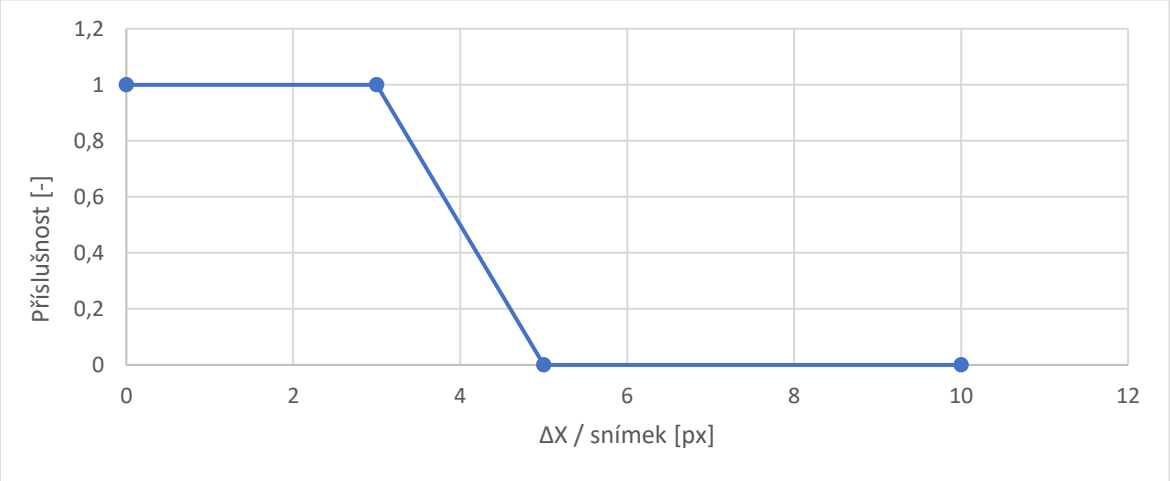
nebyl detekován další obličej na jiné straně obrazu, čímž by došlo k okamžitému nárůstu rychlosti pohybu.



Obrázek 2.10: Funkce příslušnosti pro změnu rychlosti

Z obrázku 2.10 je patrné že při nízké změně rychlosti bereme data za chybná, změna může být způsobena kolísáním detekované velikosti tváře, zatímco při vysoké změně rychlosti dochází k detekci tváře na druhé straně obrazu, což je tvář někoho jiného, kterou chceme z detekce vyloučit. Hodnoty změny rychlosti v rozmezí 5–100 jsou brány jako správně detekovaný hráč.

Pro zrušení detekce chybných pohybů uživatele bude použita druhá implementace fuzzy logiky pro detekci chybného posunu v 2-D souřadnicích. Při vysoké změně jedné ze souřadnic můžeme předpokládat, že jde o chybnou detekci nebo obličej, který zmizel a následně se objevil na vzdálenější pozici.

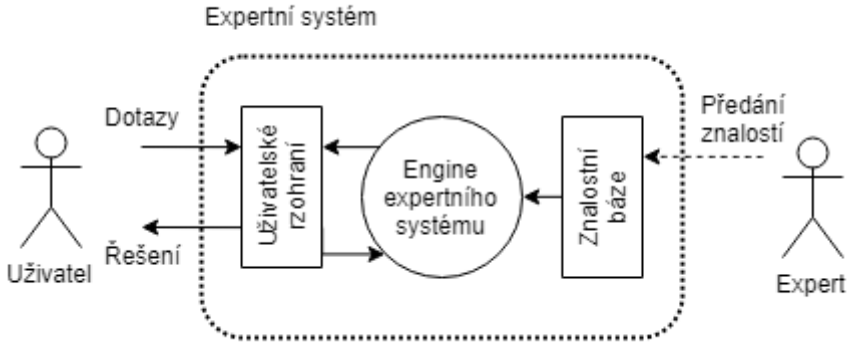


Obrázek 2.11: Funkce příslušnosti pro změnu jedné ze souřadnic

2.4.2 Expertní systém

Expertní systém ve své podstatě patří mezi slabě inteligentní systému, neboť nepoužívá vlastní řešení, ale pouze řešení poskytnuté expertem v dané oblasti ve formě znalostní databáze. Expertní systém je tedy úzce zaměřený z důvodu poskytování relevantních informací k určitému tématu. Pokud by byla znalostní databáze příliš tematicky široká, docházelo by k chybným odhadům a tím pádem chybnému chování navrženého systému. Protože je expertní systém na bázi

znalostí člověka, dochází k určité chybovosti, neboť i expert, který předává své znalosti, může opomenout některá fakta, zapomenout dodání některých dat a tím způsobí chyby v systému. Pokud navíc uživatel udělá chybu při obsluze systému, může systém poskytnout naprosto chybné řešení. Dalším problémem expertního systému je neschopnost zpracovat data v neočekávané podobě. Navíc expertní systém není schopen samostatně detekovat podivné chování uživatele, což člověk (expert) díky zdravému rozumu pozná. [12]



Obrázek 2.12: Princip fungování expertního systému

Druhou použitou implementací umělé inteligence bude vytvoření expertního systému. Systém bude navrhnut jako jednoduchá hra, kdy uživatel odpovídá na otázky typu Ano/Ne. Na základě odpovědí robot odhadne dopravní prostředek, který si uživatel myslel. V rámci návrhu je nutné vytvořit znalostní databázi pro expertní systém, tak aby pokryla co nejvíce možných dopravních prostředků a robot tak byl schopen uhodnout téměř jakýkoliv dopravní prostředek.

2.4.3 Strojové učení

Strojové učení je nejvyspělejší implementovanou formou umělé inteligence. Algoritmus strojového učení je součástí metody detectMultiScale, která je součástí OpenCV. Tato metoda přijímá 7 parametrů. Prvním je obraz, ve kterém chceme objekty detekovat, dalším je vektor, do kterého se uloží všechna detekovaná data. Následnými 5 parametry se nastavuje detekční algoritmus. ScaleFactorem definuje zmenšení každé části obrazu při každém kroku. Parametr minNeighbors definuje minimální počet detekovaných sousedů, které musí prvek mít, aby byl označen jako detekovaný. Pomocí flags nastavujeme funkční mód metody. Poslední dva parametry specifikují minimální a maximální velikost detekovaného objektu.

Pro zvolenou aplikace byla testováním určena sada parametrů – ScaleFactor = 1.3, minNeighbors = 3, flags = 0 a minimální velikost 2x2 pixely. Maximální velikost jsem nedefinoval, algoritmus nastaví automaticky maximální velikost detekované oblasti podle rozměrů obrazové matice.

3 Realizace robotického systému

3.1 Struktura

Software robotického systému se skládá ze dvou částí. První je hlavní spustitelná aplikace napsaná v C#. Ta zajišťuje vykreslení a obsluhu GUI a všechny potřebné vnitřní úlohy. Druhou částí je knihovna napsaná v C++/CLI, která zaštiťuje komunikaci s kamerou a detekcí obličeje.

Během realizace došlo k implementaci všech navrhnutých algoritmů a jejich optimalizaci. Nejprve došlo k implementování knihovny pro práci s kamerou, protože bez jejího použití by nefungoval zbytek aplikace. Následně došlo k implementování hlavního menu s možností výběru hry a hry „Kvíz“. Tato implementace trvala nejdéle, protože docházelo k úpravám algoritmů a k optimalizacím předávání dat. Když byla hra plně implementována, došlo k implementaci hry „Fotbal“. V této hře byl použit již základ ze hry „Kvíz“, pouze došlo k úpravě pravidel hry. Problémy nastaly při implementování základní fyziky míče, kdy se míč občas choval naprosto chybně. Došlo tedy k úpravě a zjednodušení algoritmů pro výpočet pohybu míče a chyba se přestala projevovat. Poslední částí implementace bylo implementování hry „Uhádnu dopravní prostředek“. Tato hra využila základu obou předchozích her a její realizace byla relativně rychlá. Po implementování všech pravidel hry došlo k drobné úpravě realizace kvůli chybnému zodpovídání otázek po zobrazení nové otázky.

3.2 Realizace knihovny pro obsluhu kamery

Knihovna pro detekování obličeje a předání dat do vizualizace je realizována v jazyce C++/CLI. Tento jazyk byl vyvinut jako nástupce Managed Extensions for C++ a slouží jako mezičlánek mezi C++ a C#, respektive umožňuje datovou výměnu mezi oběma jazyky.

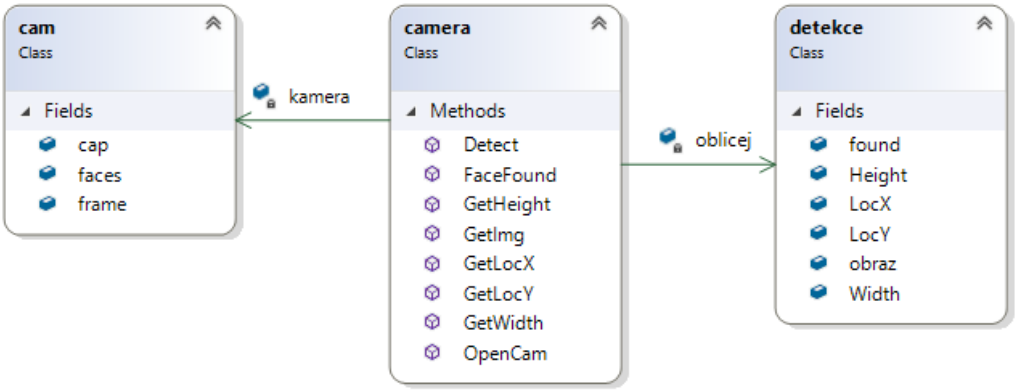
Tělo knihovny tvoří 3 třídy a to třídy cam, camera a detekce. Z důvodu toho, že CLI nepovolí mít nespravované proměnné v třídě, která je viditelná z C#, musel jsem pro objekt kamery a detekovaného obličeje vytvořit vlastní třídy. Ty už se tváří jako spravované proměnné a je možné je použít.

Třída cam si drží referenci na otevřené vstupní zařízení (kameru), aktuální snímek z kamery a soubor kaskádového klasifikátoru pro detekci obličeje.

Třída detekce si drží celočíselné kolekce, ve kterých jsou uloženy data o všech nalezených obličejích v aktuálním snímku. Krom těchto dat třída obsahuje aktuální snímek ve formě bitmapy, protože bez toho bychom nebyli schopni vykreslit bitmapu do uživatelského rozhraní.

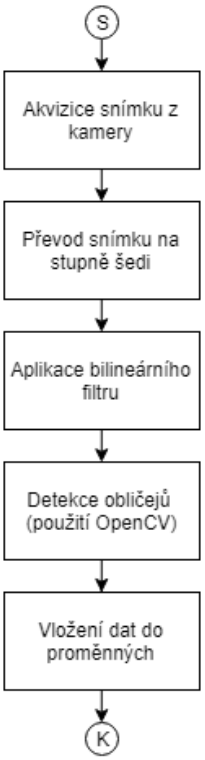
Třída camera obsahuje kompletní interface pro komunikaci s vizualizační částí. Všechny její metody jsou veřejné, protože předávají data do vizualizace. Knihovnu inicializujeme pomocí metody OpenCam, kde dojde ke spuštění kamery a nastavení její reference do proměnné kamera. Samotná detekce obličeje se provede pomocí zavolání metody Detect. Tato metoda sama o sobě

nevrátí výsledek. Pouze nám poskytuje informaci o tom, že proběhla. To, zda se povedlo detekovat obličej, zjistíme pomocí metody FaceFound. Pomocí Get metod dostaneme jednotlivé kolekce. Vyjímkou je metoda GetImg, která předá aktuální snímek převedený na bitmapu.



Obrázek 3.1: Třídní diagram knihovny

Z použitých metod je nejdůležitější metoda Detect (obrázek 3.2). Prvním krokem, který metoda provede je získání snímku z kamery. K tomuto účelu je použita opensource knihovna OpenCV, která usnadňuje práci s vizuálními daty, ať již ze snímacího zařízení nebo z uložených dat. OpenCV pracuje s obrazovými daty pomocí objektu typu Mat. Jde o matici obrazových dat v požadovaném barevném prostoru.



Obrázek 3.2: Blokové schéma metody Detect

Při načtení dat z kamery je pořízený snímek zrcadlově převrácen. Z důvodu toho, že obraz zpětně promítáme, musíme ho obrátit, jinak by vypadal nereálně a pro uživatele nepřirozeně. Po obracení obrazu převedeme obraz do odstínu šedé. Před provedením úprav vytvoříme novou matici založeném na datovém typu unsigned char, protože nám stačí pouze jedna hodnota pro každý bod v rozmezí 0 – 255. Následně pomocí původní matice a rovnice 2.1 ve dvou vnořených smyčkách vypočteme nové hodnoty pro matici. Na hotové matici obrazu v odstínech šedi se následně provede bilineární filtrace pomocí masky 3x3 pixely. V této masce jsou pixelům na osách přiřazeny vyšší váhy, aby nedošlo k přílišnému zkreslení středového bodu.

```
cv::Mat line(gray.rows, gray.cols, CV_8UC1);
for (int i = 1; i < gray.cols - 1; i++)
{
   for (int j = 1; j < gray.rows - 1; j++)
   {
      double sum = 0;
      int B1 = gray.at<uchar>(cv::Point(i - 1, j - 1));
      int B2 = gray.at<uchar>(cv::Point(i, j - 1));
      int B3 = gray.at<uchar>(cv::Point(i + 1, j - 1));
      int B4 = gray.at<uchar>(cv::Point(i - 1, j));
      int B5 = gray.at<uchar>(cv::Point(i, j));
      int B6 = gray.at<uchar>(cv::Point(i + 1, j));
      int B7 = gray.at<uchar>(cv::Point(i - 1, j + 1));
      int B8 = gray.at<uchar>(cv::Point(i, j + 1));
      int B9 = gray.at<uchar>(cv::Point(i + 1, j + 1));

      sum = B1 + 2 * B2 + B3 + 2 * B4 + 4 * B5 + 2 * B6 + B7 + 2 * B8 + B9;
      int val = sum / 16;

      line.at<uchar>(cv::Point(i, j)) = val;
  }
}
```

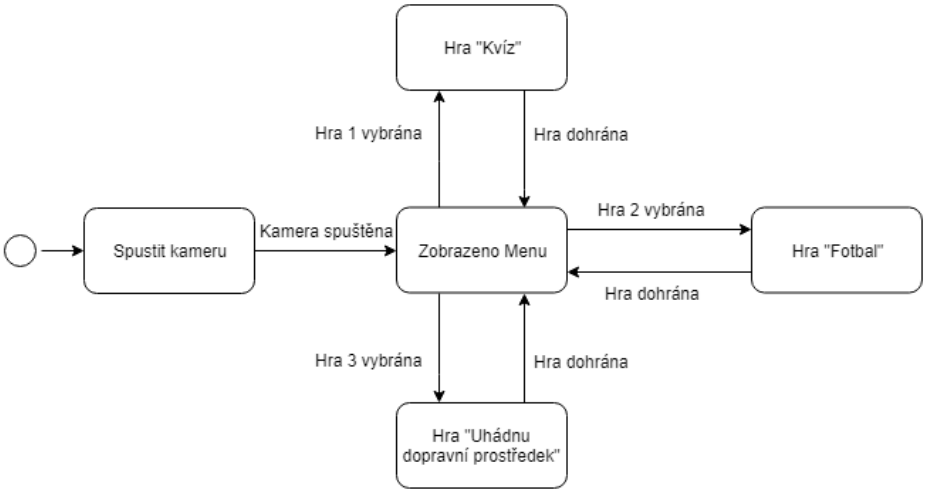
Zdrojový kód 1: Bilineární filtr 3x3 pixely

Před samotnou detekcí obličeje se vytvoří vektor typu Rect, ve kterém jsou uloženy informace o detekovaných obličejích. Dalším krokem je zavolání metody detectMultiScale, která je součástí knihovny OpenCV. Tato metoda implementuje celý algoritmus pro detektor Viola Jones. Pokud metoda detekuje alespoň jeden obličej, naplní se všechna výstupní pole třídy detekce a metoda skončí.

3.3 Realizace vizualizační aplikace

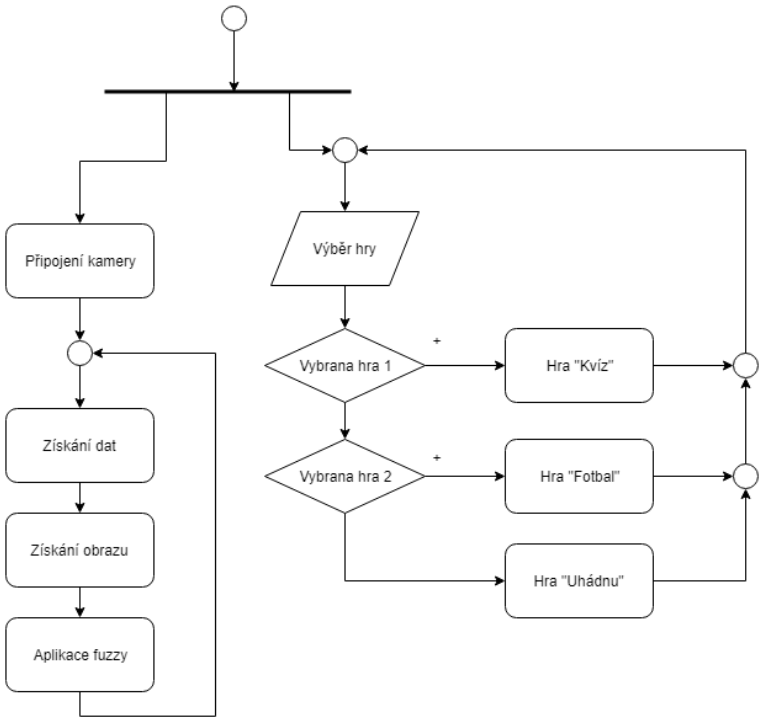
Realizovaná aplikace se skládá z čtyř částí – menu a třech jednotlivých her. V menu je umožněna volba jednotlivých her a aplikace se po dohrání hry vrátí zpět do menu. První hrou je

jednoduchý kvíz, ve hráč odpovídá na otázky týkající se elektrotechniky. Druhou hrou je jednoduchý fotbal, jehož cílem je pomocí postrkování dostat míč do branky. Poslední hrou je věštec, který uhodne, na jaký dopravní prostředek hráč myslí.



Obrázek 3.3: Stavový diagram aplikace

Ze stavového diagramu aplikace (obrázek 3.3) vidíme všechny stavy systému a definované přechody mezi nimi. Z diagramu je patrné, že systém se vždy po ukončení hry navrátí do menu, kde se přesune i po spuštění kamery.



Obrázek 3.4: Zjednodušený vývojový diagram aplikace

Zjednodušený vývojový diagram (obrázek 3.4) zobrazuje průběh celé aplikace. Z diagramu je patrné, že aplikace se po spuštění rozdělí na dvě vlákna – vizualizační a kamerové. Vlákno obsluhující kameru je spuštěno po celou dobu programu nezávisle na vizualizaci, pouze ji poskytuje data, kterých využívá vizualizační vlákno.

3.3.1 Realizace hlavního menu

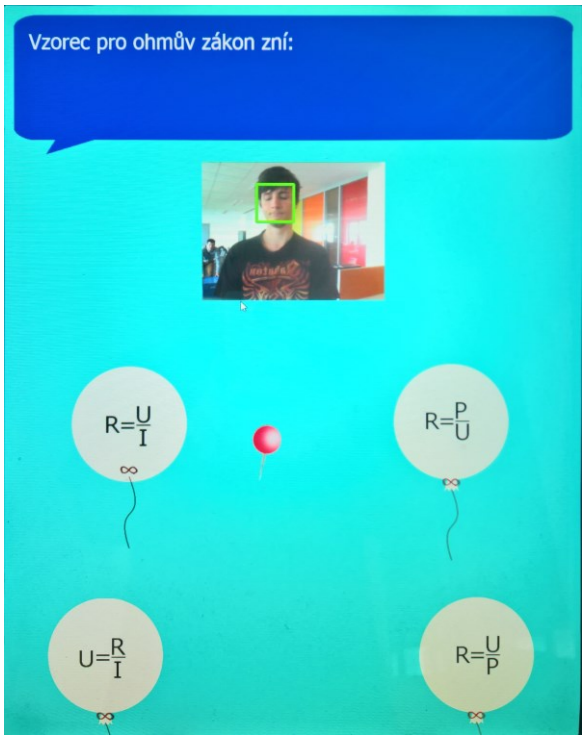
Po spuštění aplikace dojde k spuštění hlavního uživatelského menu. To se skládá ze tří jednotlivých tlačítek tvořených jako labely a jednoho labelu na informační popis. Všechny vizuální části jsou napsané v značkovacím jazyce XAML. Tento jazyk je odvozen od zápisu XML značkování, pouze je určen pro tvorbu rozhraní. Při prvním spuštění celé aplikace se oddělí vlákno DATA_READER, které pracuje s vnitřní knihovnou pro detekci obličeje a aplikací fuzzy logiky na získaná data. Toto vlákno běží po celou dobu aplikace separátně na pozadí a předává data do statických datových polí hlavního okna. Při doteku v místě labelu s příslušnou hrou dojde ke spuštění hry a po jejím skončení se navrátí zpět do menu.

Obrázek 3.5: Hlavní menu

3.3.2 Realizace hry „Kvíz“

Hra „Kvíz“ funguje na principu výběru jedné ze 4 odpovědí pohybem hlavy. Odpovědi jsou znázorněny jako balónky s obrázkem, popřípadě textem, které musí hráč propíchnout. Odpovědi se také jako balónky chovají, to znamená, že pokud hráč do balónku jen jemně ťukne, dojde k jeho odrazu. Odpověď je zvolena jako správná až po vyvinutí dostatečné rychlosti a propíchnutí balónku. Po propíchnutí dojde k vyhodnocení správnosti odpovědi a načtení další

otázky. Po zodpovězení všech otázek se uživateli ukáže jeho výsledné skóre a systém se automaticky přepne zpět na výběr hry. Vzhled hry můžeme vidět na obrázku 3.6.



Obrázek 3.6: Hra „Kvíz“

Při spuštění nové otázky dojde k vytvoření nového vlákna VALUE_UPDATER, které se stará o celou hru. Toto vlákno se stará o získání živého obrazu, přepočet pozice kurzoru (v tomto případě špendlíku), vyhodnocení vzniku odrazu nebo propíchnutí a spouští a nastavuje animace. Dalším krokem při vytvoření otázky je nastavení vnitřních proměnných určujících text otázky, všechny obrázky a správnou odpověď.

Data pro jednotlivé otázky jsou uloženy v datovém souboru dataOtaz.xml. Zde jsou uloženy všechny informace k otázce a ID jednotlivých obrázků k odpovědím. Obrázky a zvuková data jsou uložena v resource file souboru, který se při kompilaci stane součástí .exe souboru a díky tomu se všechny obrázky stanou součástí programu.

```
<Kviz id="0">
      <Question>Vzorec pro ohmův zákon zní:</Question>
      <Ans1>1</Ans1>
      <Ans2>2</Ans2>
      <Ans3>3</Ans3>
      <Ans4>4</Ans4>
      <Ok>2</Ok>
</Kviz>
```

Zdrojový kód 2: Ukázka uložení otázky

Pro získání nového obrazu se využívá synchronizačních prvků, protože nemůžeme dovolit, aby do proměnné s obrazem zapisovala najednou dvě vlákna a to vlákno DATA_READER a VALUE_UPDATER. Pro tuto synchronizaci je využito mechanizmu monitoru. Z důvodu výkonu nechceme zpomalovat vlákno VALUE_UPDATER, aby se aplikace nejevila jako zaseknutá, používáme u vlákna VALUE_UPDATER metodu Monitor.TryEnter(object lock). Tato metoda uzamkne mutexový objekt pouze pokud je volný a vrátí, zda bylo možné mutex uzamknout. Výhodou je, že pokud je mutex právě obsazen druhým vláknem, část s překreslením obrazu se přeskočí a pokračuje se dále v kódu. Důležité je nezapomenou mutex manuálně uvolnit, jinak zůstane zamčený a žádné vlákno už se do těchto částí kódu nedostane.

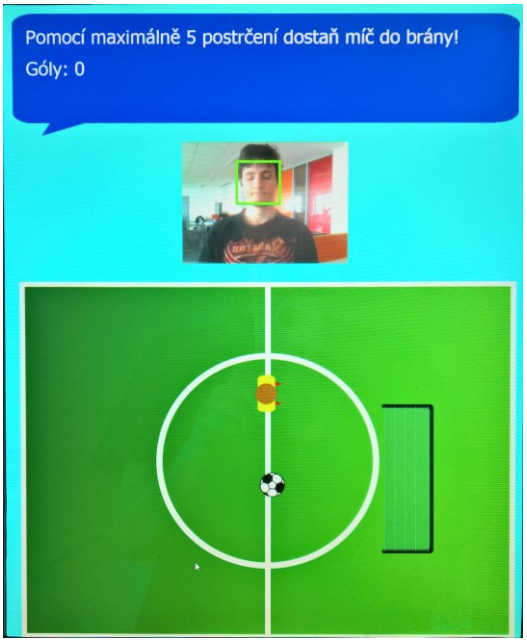
Pro animování objektů jsem využil možnost použití tříd Storyboard a DoubleAnimation. Pomocí těchto dvou můžeme animovat vlastnosti objektů uživatelského rozhraní ve WPF. Animace si nejprve pomocí DoubleAnimation nastavíme na vlastnost, kterou chceme animovat (např. výška, šířka, průhlednost…) a následně v rozmezí jakých hodnot chceme animovat. Existují dvě možnosti rozmezí. Při použití vlastnosti By určujeme relativní změnu dané vlastnosti a při použití vlastností From, To přesně definujeme, v jakém rozmezí má vlastnost měnit svou hodnotu. Důležitým parametrem je vlastnost Duration, pomocí které určíme dobu trvání animace. Následně všechny proměnné DoubleAnimation přidáme jako děti scénáře a ten následně spustíme.

3.3.3 Realizace hry „Fotbal“

Cílem hry „Fotbal“ je pomocí 5 postrčení dostat míč do brány. Hráč má šanci dát 5 gólů. Po pátém postrčení, které neskončí míčem v brancem, se připočte jeden neúspěšný pokus a míč se posune na náhodné místo na levé straně hřiště. Ve chvíli, kdy je míč v oblasti brány dojde k přičtení úspěšného vstřelení gólu a inicializaci dalšího pokusu. Po ukončení hry se zobrazí vyhodnocení hry a následně se aplikace vrátí do hlavního menu.

Aktualizace dat probíhá stejným způsobem jako u předchozí hry. Paralelní vlákno pro aktualizaci dat se stará o posun hráče po hřišti, vyhodnocování odrazu míče a vykreslení aktuálního obrazu z kamery. Dále vlákno vypočítává vektor pohybu hráče, aby se míč odrazil správným směrem od hráče.

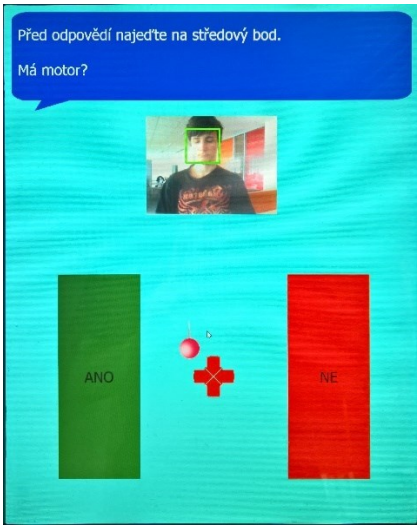
Vektor pohybu se určuje na základě posledních dvou pozic fotbalisty na ploše, následný vektor se zvětší pro simulaci kopnutí a následně se simuluje tření míče, které postupně míč zastaví. Po zastavení míče je uživateli umožněno kopnout znovu. Během odrazu míče je nutno zachovat přiměřenou realističnost – míč se odrazí od konstrukce brány a patřičně upraví svůj pohyb. V rámci toho, aby bylo možné hrát i při chybném kopu, který vede mimo plochu hřiště byly vytvořeny umělé mantinely, které zabrání fotbalovému míči v opuštění hřiště a tím k znemožnění dalšího postupu hrou. Vizuální stránku této hry můžeme vidět na obrázku 3.7.



Obrázek 3.7: Hra „Fotbal“

3.3.4 Realizace hry „Uhádnu dopravní prostředek“

Hra „Uhádnu dopravní prostředek“ je realizací expertního systému. Hráč je na začátku vyzván, aby si myslel jeden dopravní prostředky a poté odpovídá na otázky typu ano/ne. Protože v této hře stačí pro odpověď najet na jednu stranu s odpovědí, byl zde implementován mechanizmus povolení odpovědi. Hráč před odpovědí na novou otázku musí projet středovým bodem, který po projetí zmizí. Toto bylo nutné z důvodu vykreslení nové odpovědi, kdy hráč byl stále na pozici odpovědi na minulou otázku a tím došlo i k jejímu vyhodnocení. Vzhled hry i se středovým bodem vidíme na obrázku 3.8.



Obrázek 3.8: Hra „Uhádnu dopravní prostředek“

Pro účely této hry byl realizován celý expertní systém. Znalostní báze systému obsahuje 30 dopravních prostředků, které systém při očekávaném vstupu od uživatele dokáže rozpoznat. Na základě báze bylo vytvořeno 14 otázek, pomocí kterých lze dopravní prostředky rozpoznat. Jednotlivé otázky jsou pokládány v předem určeném pořadí, každé otázce je přiřazena hodnota odpovědi odpovídající 2^x, kde x je ID odpovědi (0-14) a jednotlivým dopravním prostředkům je určena hodnota součtu odpovědí a ID otázky, u které je již možno prostředek určit. Znalostní bázi můžeme vidět na obrázku 3.9. V bázi jsou odpovědi ano označeny jako A, odpovědi ne označeny jako N a otázky, na které již není třeba odpovídat jako K. Ty jsou vždy zapsány až tam, kde známe dopravní prostředek.

	Prostředek	Motor?	Jezdí po zemi?	Má pneumatiky?	Sveze více než jednu osobu?	Jezdí po kolejích	Jezdí mezi městy?	Má více než 4 kola?	Umí létat?	Jezdí na elektro	Používá se k záchraně lidí?	Musíš při jízdě pracovat rukama?	Musíš šlapat?	Jezdí v podzemí?	Má dvě kola?
0	Auto	A	A	A	A	N	A	N	N	A	K	K	K	K	K
1	Kolo	N	A	A	N	N	N	N	N	N	N	A	A	K	K
2	Nohy	N	A	N	N	N	N	N	N	N	A	N	A	K	K
3	Vlak	A	A	N	A	A	A	K	K	K	K	K	K	K	K
4	Tramvaj	A	A	N	A	A	N	A	N	A	N	N	N	N	K
5	Letadlo	A	N	A	A	K	K	K	K	K	K	K	K	K	K
6	Koloběžka	N	A	A	N	N	N	N	N	N	N	A	N	K	K
7	Loď	A	N	N	A	N	A	N	N	K	K	K	K	K	K
8	Brusle	N	A	N	N	N	N	A	K	K	K	K	K	K	K
9	Skateboard	N	A	N	N	N	N	N	N	N	N	N	K	K	K
10	Autobus	A	A	A	A	N	A	N	N	N	K	K	K	K	K
11	Trolejbus	A	A	A	A	N	N	K	K	K	K	K	K	K	K
12	Segway	A	A	A	N	N	N	N	N	A	N	A	N	N	A
13	Vznášedlo	N	A	N	A	N	N	N	N	N	N	A	K	K	K
14	Rogalo	A	N	N	N	K	K	K	K	K	K	K	K	K	K
15	Vrtulník	A	N	N	A	N	N	N	A	N	A	K	K	K	K
16	Padák	N	N	N	N	K	K	K	K	K	K	K	K	K	K
17	Lyže	N	A	N	N	N	N	N	N	N	A	N	N	K	K
18	Sáně	N	A	N	A	N	N	N	N	N	N	N	K	K	K
19	Vzducholoď	A	N	N	A	N	A	N	A	K	K	K	K	K	K
20	Motorka	A	A	A	N	N	A	K	K	K	K	K	K	K	K
21	Veslice	N	N	N	A	N	N	N	N	N	N	A	K	K	K
22	Šlapadlo	N	N	N	A	N	N	N	N	N	N	N	K	K	K
23	Plachetnice	N	N	N	A	N	A	K	K	K	K	K	K	K	K
24	Nákladní auto	A	A	A	A	N	A	A	K	K	K	K	K	K	K
25	Raketa	A	N	N	A	N	N	N	A	N	N	K	K	K	K
26	Metro	A	A	N	A	A	N	A	N	A	N	N	N	A	K
27	Běžky	N	A	N	N	N	N	N	N	N	N	A	K	K	K
28	Motokára	A	A	A	N	N	N	N	N	A	N	A	N	N	N
29	Čtyřkolka	A	A	A	N	N	N	N	N	N	A	K	K	K	K
	Hodnota odp.	1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192
	Číslo otáz.	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Obrázek 3.9: Znalostní báze hry

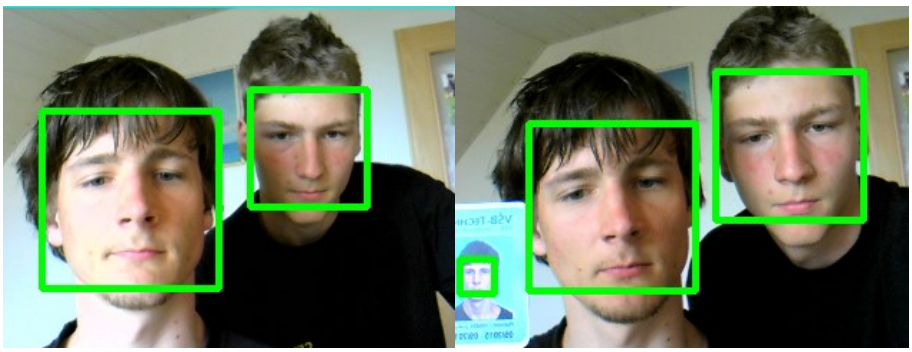
Při běhu hry dojde po odpovědi na každou otázku k vyhodnocení, zda již prostředek známe. Pokud ano, dojde k vyhodnocení hry, tj. zobrazení prostředku, který si hráč myslel a přechodu zpět do menu. Pokud ani po vyčerpání otázek není možné prostředek určit, je vyhodnocen jako neznámý.

4 Verifikace a optimalizace

Verifikace funkčnosti a optimalizace programu je důležitou součástí jak návrhu, tak i realizace softwaru. Při verifikaci jsem ověřil, že software je schopen detekovat i více obličejů a správně na ně reagovat. Optimalizací obecně rozumíme vylepšení či zefektivnění algoritmu za účelem zrychlení běhu či vylepšení výpočetních výsledků.

4.1 Verifikace detekce

Verifikaci detekce jsem prováděl za použití více osob stojících před kamerou a také vytištěného obličeje. Z obrázku 3.10 je patrné, že byly detekovány všechny obličeje. Program správně reagoval na více obličejů v obraze vybráním největšího a použití pouze jeho pozice.



Obrázek 3.10: Verifikace detekce

Při změnách osvětlení či naklonění do obličeje docházelo k problémům s detekcí obličeje, popřípadě s správným určením jeho velikosti. V těchto případech nebyly některé obličeje detekovány, popřípadě detekovány ob jeden či dva snímky. Při nízké intenzitě osvětlení je vlivem špatného rozdílu mezi světlou a tmavou částí obličeje detekce velmi problematická.

4.2 Optimalizace programu

Již během vývoje docházelo k optimalizaci, protože po napsání některých částí kódu došlo k enormnímu zpomalení běhu programu až na úroveň, kdy se jevil jako trhaný a tím pádem uživatelsky nepříjemný. Došlo i k dalším optimalizacím, například výstupy z vláken, kde jsou zápisy a čtení z proměnných byly seskupeny do jednoho místa, a ne do několika různých míst. Důležitou optimalizací bylo snížení rozlišení kamery, které přineslo snížení průměrné doby detekce z 180 ms na 66 ms.

Také v rámci optimalizace byl upraven původní návrh, kdy se mělo jednat o dva nezávislé programy a knihovnu na jeden program s knihovnou. Data mezi těmito programy byly předávány pomocí Named Pipes, což zpomalovalo program, neboť bylo nutné data převádět na formát vhodný pro přenos rourou a následně je zase převádět na originální data. Poslední provedenou optimalizací byla oprava chybné části kódu, která spouštěla dvakrát po sobě detekci obličeje namísto pouze jednoho spuštění, což prakticky ušetřilo asi polovinu detekční doby.

5 Srovnání výsledků

Pro srovnání výsledků jsem využil druhý, výkonnější počítač, který by dle předpokladu měl poskytovat lepší výsledky. Základem tohoto předpokladu je využití novějších technologií, využití více jader, z čehož vyplývá výhoda při zpracování paralelních procesů a také využití novějších pamětí s vyšší propustností. Druhá testovací sestava se skládá z procesoru Intel Core i7-7700@3,6GHz a 16 GB DDR4 operační paměti.

V tabulce 5 vidíme změřené výsledky detekční doby na referenčním a robotickém systému. V naměřených dat je patrné, že detekční doba při stejném rozlišení je přibližně dvojnásobná. Při nastavení kamery na snímkovací frekvenci 30 FPS by detekční doba neměla překročit 33 ms, abychom stihli mezi jednotlivými snímky z kamery obraz zpracovat. U referenčního systému je snadno patrné, že průměrná detekční doba je nižší než 33 ms a program tedy stíhá běžet dostatečně rychle, aby uživateli pohyb připadal plynulý a bez trhání. Na realizovaném systému je detekční doba přibližně dvakrát vyšší. Toto může být způsobo převážně rozdílem ve výkonu obou systémů. Zrychlení by bylo možno dosáhnout změnou v HW konfiguraci systémů nebo dalším snížením rozlišení. Změna rozlišení by měla za následek menší rozlišení pohybu a horší ovladatelnost.

Tabulka 5. Srovnání detekčních dob

	Doba detekce obličeje (ms)	
ID snímku	Robotický systém	Referenční systém
0	68	28
1	59	29
2	71	44
3	65	29
4	64	28
5	57	28
6	76	28
7	62	44
8	73	29
9	59	34
10	72	19
Průměrná doba detekce (ms)	66,0	30,9

Závěr

Cílem této bakalářské práce bylo vytvořit teoretický základ zabývající se robotickými systémy, interaktivitou a umělou inteligencí. Na základě těchto poznatků bylo dalším krokem navržení a realizování robotického systému.

V rámci návrhu systému byly popsány potřebné algoritmy a návrh jednotlivých částí systému. Byla zde také navržena implementace několika druhů umělé inteligence. Všechny části byly následně realizovány, optimalizovány a verifikována jejich funkčnost.

Snahou při optimalizaci bylo zrychlit běh aplikace na úroveň reálného času. I po všech optimalizacích toho dosaženo nebylo v rámci převážně hardwarových omezení. Zrychlení běhu by mohlo být docíleno přesunem některých výpočtu, které mohou být paralelizovány, na grafickou kartu, protože umí mnohem lépe pracovat s paralelním zpracováním než normální procesor. Zde je ale potřeba uvážit, zda by režie nutná pro rozdělení dat na paralelní části, následné zpracování a spojení ve výsledný celek, nebyla výpočtově náročná a tím pádem by nedošlo k žádné úspoře, či hůře k zpomalení výpočtu.

Z uživatelského pohledu byla aplikace cílena jako hra. Snahou bylo vytvořit jednoduché, příjemné a lehce pochopitelné rozhraní s intuitivním ovládáním. I při průměrné době detekce obličeje 66 ms a následném zpracování vizualizací se aplikace jeví jako plynulá, drobné trhání může být patrné při vyšším vytížení systému jinými procesy, kdy hra dostává přiřazen omezený procesorový čas.

Z hlediska robotických systémů a jejich vývoje v budoucnosti lze konstatovat, že dochází k urychlování jejich vývoje a rychlejšímu nahrazování lidské práce tou robotickou. Nevýhodou tohoto vývoje může být snížení počtu pracovních míst, ale prozatím robotické systémy stále vyžadují údržbu a software, což zpětně pracovní místa tvoří. Vývoj robotických systémů se směřuje k větší multifunkčnosti, systémům, které dokáží automatizovat vlastní údržbu, učit se a eliminovat vlastní chyby a poruchy. Důležitou součástí vývoje těchto systému je také vývoj elektroniky, mechatroniky a softwarového inženýrství, protože jde o důležité obory, bez kterých by nebyl možný růst efektivity, výkonu a schopností robotických systémů.

Použitá literatura

[1] HOOPER, Rich. Industrial Robots. *Learn About Robots* [online]. [cit. 2018-03-29]. Dostupné z: http://www.learnaboutrobots.com/industrial.htm

[2] HOOPER, Rich. Research Robots. *Learn About Robots* [online]. [cit. 2018-03-29]. Dostupné z: http://www.learnaboutrobots.com/research.htm

[3] HOOPER, Rich. Entertainment Robots. *Learn About Robots* [online]. [cit. 2018-03-29]. Dostupné z: http://www.learnaboutrobots.com/entertainment.htm

[4] DOWLING, Kevin. *Power Sources for Small Robots* [online]. January 1997 [cit. 2018-03-29]. Dostupné z:

http://www.cs.cmu.edu/afs/cs.cmu.edu/Web/People/motionplanning/papers/sbp_papers/integrated1/dowling_power_sources.pdf

[5] SHRUM, L. J. a Yuping LIU. What is interactivity and is it always such a good thing?. In: Journal of Advertising [online]. 2002 [cit. 2018-03-29]. Dostupné z: http://www.yupingliu.com/files/papers/liu_shrum_interactivity.pdf

[6] TURING, A. M. I.—COMPUTING MACHINERY AND INTELLIGENCE. Mind. 1950, LIX(236), 433-460. DOI: 10.1093/mind/LIX.236.433. ISSN 0026-4423. Dostupné také z: https://academic.oup.com/mind/article-lookup/doi/10.1093/mind/LIX.236.433

[7] PRIDDY, Kevin L. a Paul E. KELLER. *Artificial neural networks: an introduction*. Bellingham, Wash.: SPIE Press, c2005. ISBN 08-194-5987-9.

[8] Multi-Layer Neural Network. In: Wikimedia Commons [online]. 2015 [cit. 2018-03-29]. Dostupné z:

https://commons.wikimedia.org/wiki/Category:Artificial_neural_network#/media/File:Multi-Layer_Neural_Network-Vector-Blank.svg

[9] ZHANG, Weixiong. *State-Space Search Algorithms, Complexity, Extensions, and Applications*. New York, NY: Springer New York, 1999. ISBN 14-612-1538-2.

[10] GORI, Marco. *Machine Learning: A Constraint-Based Approach*. Morgan Kaufmann, 2017. ISBN 0081006705.

[11] SZELISKI, Richard. Computer Vision: Algorithms and Applications. London: Springer, 2010. Texts in computer science. ISBN 978-1-84882-935-0.

[12] Expert System. *IGCSE ICT* [online]. [cit. 2018-04-05]. Dostupné z:

https://www.igcseict.info/theory/7_2/expert

Seznam příloh

Součástí bakalářské práce je DVD.

Seznam příloh na DVD

V adresáři „Zdrojový kód“ je obsažen kompletní zdrojový kód k softwarové aplikaci.

V adresáři „Spustitelný program“ je překompilovaná aplikace určená pro spuštění.

V souboru „ZmerenaData.xlsx“ jsou poskytnuta data změřená při porovnávání výsledků mezi robotickým systémem a referenčním systémem.

V souboru „PokynyKeSpusteniAKompilaci.txt“ jsou pokyny, dle kterých lze aplikaci zkompilovat a následně i spustit. Soubor obsahuje informaci o všech potřebných knihoven a frameworků.